REMARKS

Claims 1 to 34, 42 to 44, 49, and 53 to 60 are pending. Claims 1, 30, and 49 are independent. Favorable reconsideration and further examination are respectfully requested.

In the Office Action, the claims were rejected under the first and second paragraphs of 35 U.S.C. §112 for containing features that were allegedly unclear and that were allegedly not enabled by the specification. Without conceding the propriety of the rejections, Applicants have removed those features from the claims. Also, as shown above, Applicants have amended the independent claims to clarify that the feature information comprises an address of the remote computer and that the address comprises a known address that is stored prior to installation of the device in a system. Support for this language can be found, e.g., on page 8, lines 8 to 12 of the specification. In particular, the specification states

> Accordingly, whenever a device 14 is initially installed and powered-up, registration is automatic; as device registration status process 36 will communicate over distributed computing network 16 and check remote computer 12 to determine if device 14 is registered on database 32.

More specifically, the device could not check remote computer 12 at initial installation and power-up if it did not know the address of remote computer 12 at initial installation and power-up, which means that the address had to have been stored prior to installation of the device.

The Office Action also objects to the term "self-describing computer language". In particular, the Examiner has indicated that he is unclear of its meaning. Applicants respond by submitting that the term "self-describing computer language" is a term that is clearly understood in the art as evidence by the attached publication entitled "Mainstreaming XML-Based Enterprise Applications" (see, e.g., page 2 thereof). For example, one type of self-describing computer language, such as XML, contains tags that define data being transmitted. Applicants therefore submit that any computer language is not self-describing, and that the Office Action's interpretation of a self-describing computer language as any computer language is incorrect.

In view of the foregoing, Applicants respectfully request withdrawal of the rejections under the first and second paragraphs of 35 U.S.C. §112.

Turning to the art rejections, claims 1 to 46 and 49 to 52 were rejected under 35 U.S.C. §102(e) over U.S. Patent No. 6,560,656 (O'Sullivan) and under §102(e) over U.S. Patent No. 6,012,088 (Li); claims 2, 4, 31 and 33 were rejected over O'Sullivan in view of U.S. Patent No. 6,834,298 (Singer); claims 6 and 35 were rejected over O'Sullivan in view of U.S. Patent No. 5,586,254 (Kondo); claims 11 to 13 were rejected over O'Sullivan in view of U.S. Patent No. 6,686,838 (Rezvani); claims 14 to 17 and 38 to 41 were rejected over O'Sullivan in view of U.S. Patent No. 6,230,199 (Revashetti); claims 18 to 21, 24 to 26, and 42 to 44 were rejected over O'Sullivan in view of U.S. Patent No. 6,415,023 (Iggulden); claims 22, 23, 45 and 46 were rejected over O'Sullivan in view of Iggulden and Revashetti; and claims 27 to 29 were rejected over O'Sullivan in view of U.S. Patent No. 6,446,192 (Narasimhan). As shown above, Applicants have amended the claims to define the invention with greater clarity. In view of these clarifications, withdrawal of the art rejections is respectfully requested.

Amended independent claim 1 defines a method performed by a computer program to register the device with a remote computer. The method includes obtaining feature information stored for the device. The feature information includes information that is specific to an instance of the device. The feature information includes an address of the remote computer. The address comprises a known address that is stored prior to installation of the device in a system. The method also includes registering the device with the remote computer by transmitting the feature information to the remote computer at the known address using a self-describing computer language. The method is performed automatically when the computer program runs, and the method does not require manual invention.

The applied art is not understood to disclose or to suggest the foregoing features of claim 1. In particular, the art does not disclose or suggest that the feature information stored for a device includes an address of a remote computer, that the address comprises a known address that is stored prior to installation of the device in a system, and that the method is performed automatically when the computer program runs and does not require manual intervention.

As previously explained, O'Sullivan describes what happens when a device joins a network. Upon joining the network, in O'Sullivan, the device conducts a discovery process. In this discovery process, the device broadcasts, over a network, a packet that contains a code for use in communicating with the device. A discovery server 314 receives the broadcast and passes a reference to a lookup service 312 to the device, which enables the device to register itself with a Djinn (see, e.g., column 6, lines 47 et seq. of O'Sullivan). In response, the device registers its services with the lookup service and also registers information that may be used to communicate with the device (see, e.g., column 6, line 52 and column 8, lines 8 to 12).

Thus, unlike the invention of claim 1, in O'Sullivan, the address at which registration takes place is clearly not a known address that is stored prior to installation of the device in a system. Applicants also reiterate that O'Sullivan does not describe transmitting feature information using a self-describing computer language. Rather O'Sullivan describes the use of JAVA in O'Sullivan's distributed computing environment. JAVA is platform-independent, which means that the same commands and structures can be used on different platforms. JAVA is not self-describing in the context of the subject application, in that self-describing languages, such as XML, can carry different types of information and also describe that information.

Li describes an Internet access device that is able to communicate with a remote device on the Internet, obtain a configuration file, and register with the remote device. As clearly explained, e.g., in column 9, lines 20 et seq. and column 11, lines 8, et seq., Li's registration process is not automatic and requires manual intervention (unlike the method of claim 1). Furthermore, Li does not store feature information comprising an address of the remote computer, where the address comprises a known address that is stored prior to installation of the device in a system, and use that address to connect to the remote device. Instead, Li describes a system in which a customer provides an address of an ISP, which then provides the configuration file to the device. Finally, Li does not disclose or suggest transmitting feature information to a remote computer using a self-describing computer language.

The remaining art of record has been reviewed and is not understood to disclose or to suggest anything that would remedy all of the foregoing deficiencies of O'Sullivan and Li vis-à-vis claim 1. Accordingly, claim 1 is believed to define over the art.

Amended independent claims 30 and 49 are readable medium and apparatus claims, respectively, that roughly correspond to claim 1. These claims are believed to be patentable for at least the same reasons noted above with respect to claim 1.

Each of the dependent claims is also believed to define patentable features of the invention. Each dependent claim partakes of the novelty of its corresponding independent claim and, as such, has not been discussed specifically herein.

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific rejection, issue or comment does not signify agreement with or concession of that rejection, issue or comment. In addition, because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper, and the amendment of any claim does not necessarily signify concession of unpatentability of the claim prior to its amendment.

In view of the foregoing amendments and remarks, Applicants respectfully submit that the application is in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicants' undersigned attorney can be reached at the address shown below. All telephone calls should be directed to the undersigned at 617-521-7896.

Please apply any fees or credits due in this case, which are not already covered by check, to Deposit Account 06-1050 referencing Attorney Docket No. 11333-013001.

Respectfully submitted,

Date: Monday, July 18, 2005

[illegible]

Paul A. Pysher
Reg. No. 40,780

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

Patricia Seybold Group

Trusted Advisors to Customer-Centric Executives

Mainstreaming XML-Based Enterprise Applications

Using Oracle XML DB to Manage Financial Information within a Global Banking System

By Geoffrey E. Bock
November 2003

Prepared for Oracle Corporation

210 Commercial Street, Boston, MA 02109 • Phone 617.742.5200 • Fax 617.742.1028 • www.psgroup.com

Table of Contents

Illustrations

Mainstreaming XML-Based Enterprise Applications

Using Oracle XML DB to Manage Financial Information within a Global Banking System

By Geoffrey E. Bock, Senior Vice President, Patricia Seybold Group

Prepared for Oracle Corporation

Netting It Out

XML has enormous potential for the design and development of enterprise applications. Application developers can use XML to readily adapt the organization and structure of an underlying content repository to changing business situations. But this flexibility has costs. Storing and retrieving XML elements within XML documents is not as efficient as, say, using SQL to access data within a relational database management system.

As a vendor with world-class resources for developing and deploying industrial-strength databases, Oracle has released a database-driven product that encompasses the flexibility of XML with the efficiency of SQL. Oracle XML DB, an integrated set of information management capabilities within the Oracle 10*g* Database, features a novel XML/SQL duality: the ability to perform SQL operations on XML data, as well as XML ones on SQL (relational) data. On one hand, Oracle XML DB provides native support for storing and retrieving XML elements from XML documents. On the other hand, it continues to support SQL and provides the expected benefits of Oracle's object relational database management platform. Oracle XML DB stores information within the Oracle database and represents underlying data 'dually,' both as sets of XML elements within XML documents and as cells within relational tables.

Temenos, an Oracle business partner, produces T24, a suite of enterprise banking applications. T24 is based on a single database platform and a common security management system. T24 relies on a multi-value data model to manage the resulting complex data structure, where separate data views are comprised of easily transformed data sets. The nested data groups can be most succinctly expressed as extensible sets of XML elements, validated by an associated XML Schema documents, and easily stored in an XML database. Oracle XML DB provides the underlying repository and features scalable system performance on sample benchmark tests.

Oracle XML DB, we believe, demonstrates the enterprise-class scalability of the XML/SQL duality as well as the commercial viability of a native XML database.

The Prospects of XML for Managing Enterprise Information

The Promise of Extensible, Self-Describing Data

A Generic Syntax for Markup

XML, the eXtensible Markup Language, has enormous potential for the design and development of enterprise applications. As the W3C-endorsed standard for document markup, XML defines a generic syntax to describe data with simple yet flexible tags included within documents.

XML creates self-describing documents—called XML documents—where the tag-sets themselves are infinitely extensible. XML tags can appear in any order within a document and can contain any text-string as data. Taken together, an XML tag-set and the related text-string constitute an XML element. As a metamarkup language, application developers create XML elements as they need them—either by adding new text-strings to existing tag-sets or by defining new tag-sets on-the-fly to accommodate unforeseen situations.

Blending Structured and Unstructured Content

An XML document can contain any number of XML elements that can be used within different business situations. XML provides a unified model for managing both the well-structured content of numeric tables as well as the unstructured content encompassing text-oriented memos, reports, and other business documents.

XML includes mechanisms for managing XML elements within information domains. An XML application constitutes the predefined tag-sets where individuals or organizations agree *a priori* to apply certain markup to an identified domain—such as medical records, banking transactions, electronics products, musical scores, or cooking recipes, to name but a few. The markup permitted in a particular XML application is represented as a schema, which also is defined as an XML document.

Furthermore, when there is no *a priori* semantic agreement, an XML parser can parse XML documents to dynamically identify their structure.

XML Schema

XML Schema, another W3C standard, specifies the structure, content, and certain semantics for a set of XML documents. An XML Schema includes the valid tag-sets, the constraints placed on XML elements, and the specifications for data types, derivation, and inheritance—all factors that are essential for building content-rich enterprise applications.

As a result, an instance of an XML Schema provides the semantics for describing XML elements within a set of XML documents. An enterprise application first accesses the XML Schema for the document set and then accesses the related XML documents that contain the XML-elements.

Benefits for Enterprise Applications

XML has a number of benefits for enterprise applications. XML separates the data itself from the display and application logic. An XML document, for example, can appear both as a richly-formatted report on a full-screen desktop device and as an abbreviated list on a small-screen PDA display. Application developers can develop applications once and then provide separate templates (or style-sheets) that output to various display devices.

Moreover, XML is rapidly emerging as the preferred medium for exchanging information among disparate applications. The same XML document that defines a customer's problem report can be used to exchange information among a CRM application where it is created, an ERP application which credits the customer's account, and the field service tracking system which automatically schedules the service call.

Finally, XML is a very flexible way to store information. Application developers are not locked into a fixed schema of predefined relational database tables. Rather, they can easily extend their ability to manage both structured and unstructured information in light of changing business events simply by adding new XML-tags to their data sets.

Meeting Enterprise-Class Scalability and Robustness

The Unfulfilled Potential

But this flexibility has costs. Storing and retrieving XML elements within XML documents is not as efficient as using SQL to access data within a relational database management system (RDBMS). Managing information as XML raises three issues related to enterprise scalability and robustness.

RETRIEVAL SPEED. XML is not designed for fast information retrieval. SQL is.

In particular, an RDBMS is optimized for using SQL to query large datasets, arranged as rows and columns in predefined relational tables. An enterprise application can quickly query a table within a large dataset to find a value. The application can also use specialized hardware to accelerate database performance when required.

Querying XML-tagged data, by comparison, usually requires the overhead of file system access. For small data sets constituting hundreds of individual records (defined as rows in a table or lines in a document), performance is not necessarily an issue. For large data sets constituting thousands of records and/or when operating in a high-speed, transaction-oriented environment, application developers have to be concerned about the noticeable performance degradation of file system access.

CHARACTERS FOR DATA STORAGE AND TRANSMISSION. XML is a verbose method for exchanging data over a network. XML elements include pairs of tags and data values, both coded as human-readable alphanumeric strings. This adds additional characters for data storage and transmission—the characters required for the tag-set descriptors. SQL, by comparison, is a sparse and efficient method. Data are simply defined as the values of cells within a table and rapidly exchanged among networked applications.

DATATYPES. XML elements are not defined as native datatypes. Rather an enterprise application must parse the XML tags and interpret the values before beginning to process them. SQL, by comparison, includes separate data types—such as character, numerical, time and date/time, binary, and floating point—which an enterprise application can automatically process. The additional steps required for converting XML-tagged data into predefined data types degrade the performance of an enterprise application.

The Need for a Native XML Database

We are on the horns of a dilemma. Using XML as a data format for enterprise application development promises extensibility and flexibility, but at a cost—degraded performance and the lack of scalability of an enterprise-class environment.

We must face the fact that, at present, enterprise applications work most efficiently when structured data are stored within a RDBMS. Enterprise applications rely on SQL as a ubiquitous query language. An RDBMS provides an efficient mechanism for an enterprise application to access and retrieve the underlying data. In addition, Enterprise content management systems—specialized enterprise applications designed to management unstructured business information—usually store their metadata in RDBMS tables and reference the files containing the unstructured content that, in turn, are stored within the filing system of a host platform's operating system.

The downside, of course, is that a RDBMS requires the database tables and the related schemas to be defined up-front as part of an application design process. Once defined, these tables and schemas are difficult to enhance and extend. As a result, enterprise applications are fixed in time and cannot easily adapt to changing business requirements, such as the need to add additional data to an existing relational table.

Ideally, we would like to have a database-driven capability that encompasses the best of both worlds—the flexibility of XML and the efficiency of SQL. As a vendor with world-class resources for developing and deploying an industrial-strength database, Oracle has released just such a solution.

The Oracle Solution: Featuring the XML/SQL Duality

Storing XML Elements within an Object Relational Database

Oracle XML DB, an integrated set of information management capabilities within Oracle 10*g* Database, breaks new ground for managing XML within enterprise applications. Oracle XML DB features the XML/SQL duality where it supports both an XML-driven and a SQL-driven data model.

Oracle XML DB combines the best of both worlds. On one hand, it provides native support for storing and retrieving XML elements from XML documents. On the other hand, it continues to support SQL and provides the expected benefits of Oracle's object relational database management system (ORDBMS). This database stores information within its own internally-defined format and represents underlying data both as sets of XML elements within XML documents and as cells within relational tables.

Let's take a closer look at the competitive capabilities, application-level features, and business benefits of Oracle's XML/SQL duality.

Native Support for the XML Data Model

Oracle XML DB provides native support for the XML data model and its underlying programming constructs. It recognizes the hierarchical organization of XML-defined content. It includes built-in capabilities to parse XML elements, stored within XML documents, which, in turn, are represented as nested files and folders within a filing system. It recognizes the namespaces stored within XML documents as separate XML data items. Consequently, Oracle XML DB features the foundations for managing XML elements within high-performance enterprise applications.

XML SCHEMA. Oracle XML DB incorporates the W3C XML Schema standard into the underlying storage mechanism of the Oracle database. This means you can register any standard XML Schema in Oracle and thereafter use that schema as a 'template' to define and constrain XML documents.

XML SCHEMA CACHING. Oracle XML DB supports extensible schema management and overcomes the barrier of using XML Schema to validate and parse XML elements. An enterprise application can efficiently read, write, and process information stored within XML documents.

Oracle XML DB includes a schema cache. When an XML Schema is registered with the database, it is loaded into the Oracle XML DB Schema cache along with all of the meta-data required to map between the XML elements and on-disk representations of the data.

This means that, once the XML Schema has been registered with the database, no additional parsing or validation of the XML Schema is required before it can be used. The schema cache is shared by all users of the database. Whenever an Oracle XML DB operation requires information contained in the XML Schema, it is able to access the required information directly from the cache, which, in turn, accelerates performance.

NATIVE XML DATATYPE. Oracle provides a native XML datatype, at par with other standard database types, with one distinction: the XML Type supports alternative storage models. The XML data within an XML document can be shredded into component XML elements and attributes, left as a contiguous block of text, or stored as some combination of these two.

XML QUERY LANGUAGES. Oracle XML DB provides native support for XML query languages—XPath (including SQL2003-style SQL/XML access) and XQuery.[1] It incorporates the query processing tools to parse, generate, and interrogate the contents of

[1] There is a tight relationship between these two XML query languages. Oracle XML DB supports a draft version of W3C XQuery and will support the reference version once it is finalized by the W3C.

As the W3C working draft for the XQuery 1.0 specification notes, "XQuery Version 1.0 is an extension of XPath Version 2.0. Any expression that is syntactically valid and executes successfully in both XPath 2.0 and XQuery 1.0 will return the same result in both languages." See http://www.w3.org/TR/2003/WD-xquery-20030502/.

a query to a set of XML documents. An enterprise application can rapidly query a tree structure representing nested set of XML documents within a hierarchically-organized filing system.

XSLT STYLESHEETS. Oracle XML DB uses the template rules and other formatting elements that appear within the Extensible Stylesheet Language Transformations (XSLT) style-sheets. It includes an XSLT-based transformation engine to automatically transform XML-tagged documents into multiple display formats, store the transformed renditions generated by XSLT style-sheets, and deliver content in the appropriate formats for various devices. Consequently, companies can manage content destined for screen displays, ranging from full-screen Web browsers to small-screen PDAs, through a single set of XML-tagged documents.

XML REPOSITORY. Oracle XML DB provides a native XML repository that allows XML documents to be organized and accessed as files/folders in the Oracle database. This 'hierarchical' organization is most suitable for content-oriented XML–such as technical manuals–which are often organized as sub-sections, sections, chapters, and entire books. This native XML repository supports WebDAV and document versioning. This repository is also SQL-queryable.

DOM FIDELITY. XML has a number of constructs that are difficult to capture in a standard relational database paradigm. For instance, ordering nodes in a hierarchy is possible in XML but not in a relational model. As a result, a brute-force mapping of XML to the relational database paradigm results in a loss of fidelity, where such information as ordering namespaces is lost.

Oracle XML DB preserves the full fidelity of an XML document's underlying Document Object Mode (DOM). This means that, as programs manipulate the XML data, the processes of storing, retrieving, and accessing them preserves the order of individual XML documents, elements, and namespaces within a nested tree-structure of an information hierarchy.

Native Support for the SQL Data Model

As the other side of the XML/SQL duality, Oracle continues to provide native support for the SQL data model and its underlying programming constructs within Oracle 10*g* Database. This includes:

- Object relational database access that is optimized for performance
- Predefined database schemas and fixed columns within relational tables
- The ability to manage sets of the relational tables that serve as containers for structured data

In addition to the Oracle XML DB functionality, Oracle 10*g* Database features interactive query and reporting capabilities, relying on SQL99 (with SQL2003 support on the way) as the standard query language. XML elements registered through XML Schema can be displayed in relational tables and accessed through SQL. Data elements written to SQL tables can be transformed into appropriately defined XML elements and incorporated into XML documents.

Thus, application developers can continue to use SQL as their query language for accessing enterprise data and for developing specific reports. Application developers can rely on Oracle Database to provide the scalable and highly-available environment for reading, writing, retrieving, and managing the on-disk storage of underlying application data.

Multiple Protocol Handlers

With the XML/SQL duality, applications can access the underlying information both as structured data (arranged in relational database tables) and as semi-structured content (organized as XML elements within nested sets of XML documents). Enterprise applications can use separate information access protocols for retrieving data sets and displaying them on individual output devices.

Oracle XML DB supports both data-oriented access protocols and content-oriented access protocols. It provides both tightly-coupled and loosely-coupled access to the underlying data, stored within the database-driven repository.

DATA-ORIENTED ACCESS PROTOCOLS. Oracle XML DB can utilize Oracle Net, Oracle's SQL-centric client/server protocol running on TCP/IP networks (as well as on other transport mechanisms) for tightly-coupled interactions. This protocol is optimized for high-speed data transfers of structured data between clients and servers. Oracle XML DB also includes support for JDBC and ODBC as popular standards for client/server data exchange within heterogeneous network environments.

CONTENT-ORIENTED ACCESS PROTOCOLS. Oracle XML DB includes built-in support for HTTP, FTP, and WebDAV. Oracle XML DB enables Web browsers, clients running desktop applications, and remote servers running on TCP/IP networks to directly access the XML-tagged content for loosely-coupled, content-oriented interactions. As they are utilizing an underlying database, these protocols provide capabilities for high-efficiency information transfers of unstructured content.

Popular API Access

With the XML/SQL duality, application developers can use various APIs to query the contents of the underlying database. They can optimize their query and retrieval syntax that best meets their specific requirements.

For data-oriented access, application developers can continue to use a relational model based on SQL (including Oracle's highly-optimized PL/SQL) to incorporate the capabilities of existing enterprise applications.

For content-oriented access, application developers can use the XML Repository to store XML documents, coupled with XML Query or SQL2003 SQL/XML to query the documents. They can also rely on DOM to secure dynamic access to remote content.

Focusing on Performance and Scalability

As described above, Oracle XML DB manages XML as a native data-type. Oracle XML DB provides native support for the XML data model and delivers the flexibility of both data-oriented and content-oriented access to XML without the performance and scalability tradeoffs that are often associated with XML.

Application developers can use Oracle XML DB to build and deploy a high-performance, enterprise-scale application environment. Let's look at the experience of one of Oracle's business partners, a company that is now poised to reap substantial rewards from its investment in Oracle XML DB to deliver a flexible and scalable enterprise application infrastructure.

Temenos T24: Using XML DB in a Global Banking Application

Competing within the Financial Services Industry

A Worldwide Provider

Headquartered in London and operating from over 22 regional offices around the world, Temenos develops and sells banking systems into a competitive, global marketplace. Temenos T24, its core product, addresses all aspects of banking activities—including retail banking, private banking and wealth management, treasury, corporate banking, and general operations.

Temenos produces, installs, and supports the mission-critical, back-office banking applications and links them to customer-facing, front-office applications. Formerly named Globus, T24 provides the enterprise application infrastructure that banks use to run their core business operations—accessing, crediting, and debiting the multiple accounts, representing different financial products, that customers maintain with their financial institutions.

Temenos has over 300 customers worldwide, operating in more than 100 countries. Well-recognized banks and financial services firms such as ING, Credit Suisse Private Banking, Commerzbank, Standard Bank, Nedbank, Mellon Bank, Julius Baer New York, Societe Generale, Merrill Lynch, and American Express have installed and are using T24 for managing their banking transactions. Temenos is also an Oracle business partner.

An Integrated View of Business Activities

T24 provides an integrated view of all the business that a customer does with a bank, spanning multiple accounts, multiple lines of business, and multiple locations around the world. Temenos incorporates five financial application areas into T24 to support retail banking, private banking and wealth management, treasury, corporate banking, and general operations.

For example, a company concerned about efficient financial operations will often rely on a single institution for both its corporate banking and its treasury activities. Senior financial managers expect an integrated view between the company's commercial loan and cash management positions that are managed by the bank's corporate banking group on one hand, and the company's position in capital markets, including its exposure to swaps and futures/options, that are managed by the bank's treasury group on the other hand.

Technical Capabilities

Platform and Development Environment

T24 is based on a single database platform and a common security management system that unify the complete range of an institution's business, as shown in Illustration 1. It has a separate data presentation layer, where it supports multiple output devises and channel services—including Web browsers, desktop Windows-based devices, call centers, mobile and wireless, and XML-formatted messaging. T24 also features a unified set of support modules, enabling Temenos to deliver an extensive application development environment for banking applications. T24 features extensively-documented APIs, a full parameter set, and an application developer's toolkit.

For individual banks and their customers, T24 ensures the ongoing, integrated management of financial activities. Temenos can efficiently customize and adapt T24 to a bank's operational environment.

How T24 Works

T24 is a contract-based system that continually updates the records in the underlying database. Every transaction that a user enters is captured as a database record, simplifying application design and development and ensuring ongoing monitoring of transaction-related events.

End-users only need to query the database once to retrieve results, a factor that contributes to high performance and rapid through-put. T24 propagates the end-users' inputs to the specific modules of the banking application and concatenates the results of a query into a single table display, dynamically reformatting the fields within the table to match the query.

A Multi-Value Data Model

T24 relies on a multi-value data model to manage the resulting complex data structure, where separate data views are comprised of easily transformed data sets.

The multi-valued data model allows the design of the database to reflect the organization and structure of business data in a consistent, efficient manner. All information about an account is held in a single row (or list) in a database table. This means that when end-users query the database to retrieve the information they need, they find all of the data in a single row (or list). If the data were stored in classic relational terms, then there would be multiple queries against multiple tables.

A multi-value data model is characterized by ability to store repeating, nested data groups, which can be continually extended in light of business activities. Rather than writing new data into predefined cells of a relational database table or adding additional tables to a database, the nested data groups can be most succinctly expressed as extensible sets of XML elements, encompassing defined tags and field/value pairs. These elements are validated by an associated XML Schema document.

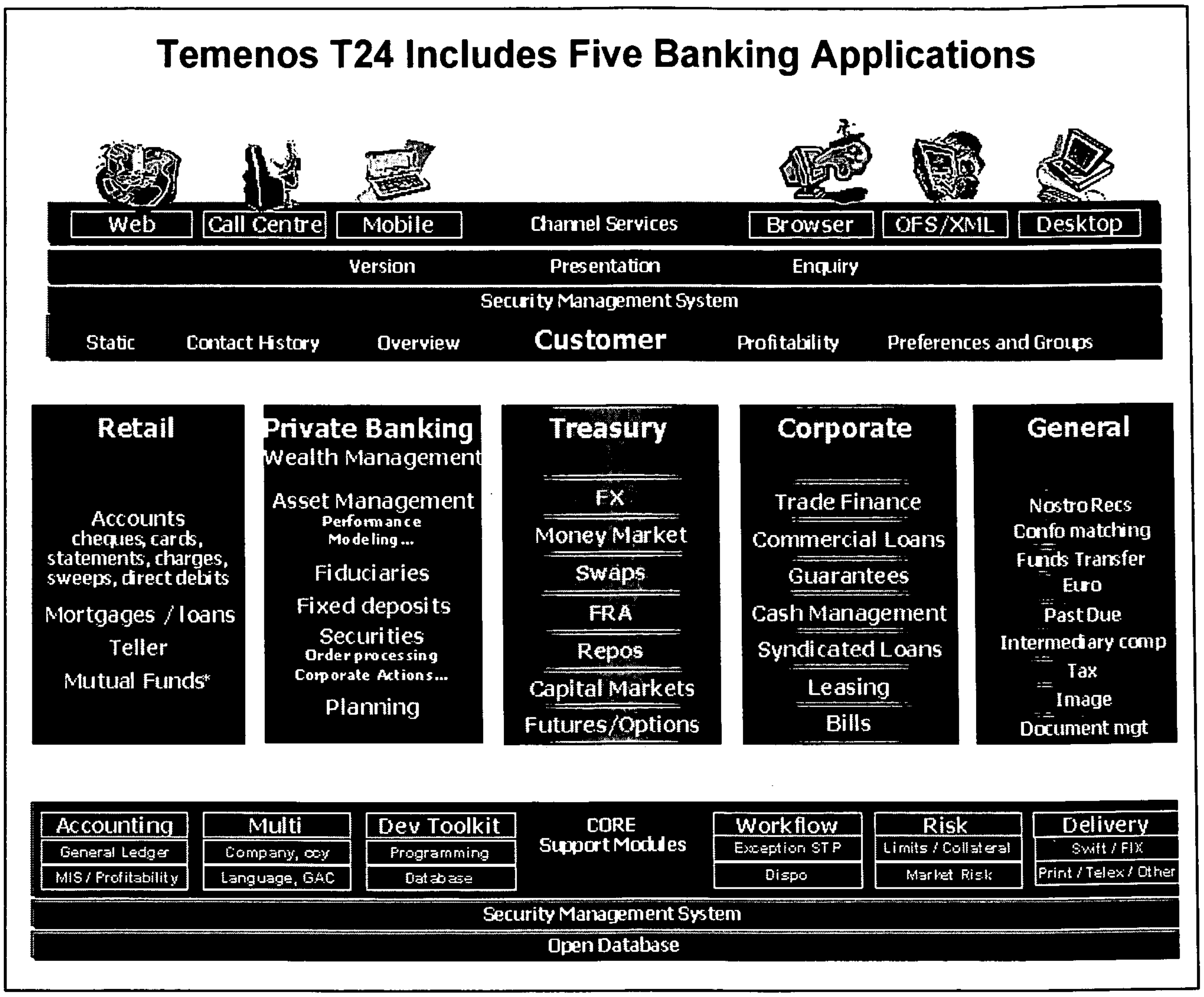

Illustration 1. Temenos T24 includes five banking applications—retail banking, private banking and wealth management, treasury, corporate, and general operations—organized around a common platform and built on a single database infrastructure. T24 features a single display environment as well as a unified security infrastructure.

The Role for Oracle XML DB

T24 stores data as XML elements and relies on XML Schema for organizing, structuring, and validating the content of its multi-value data model. To deliver on the promise of high-performance throughputs, Temenos combines the flexibility and extensibility of an XML-based multi-value data model with the scalability, reliability, data accessibility, and industrial-strength performance of SQL-driven banking applications.

This is where Oracle XML DB, with its core support for the XML/SQL duality, provides a compelling solution. As shown in Illustration 2, end-users interact with T24 at the presentation layer. As they enter information online, they produce XML elements which are validated by an XML Schema document and written directly into the underlying database. T24 banking applications reside on the tServer, a mid-tier application server. As end-users query the application, the relevant modules within the banking application produce SQL calls to the database. The database returns results as SQL statements within a report, which are then appropriately formatted for the display device and presented to the end-users.

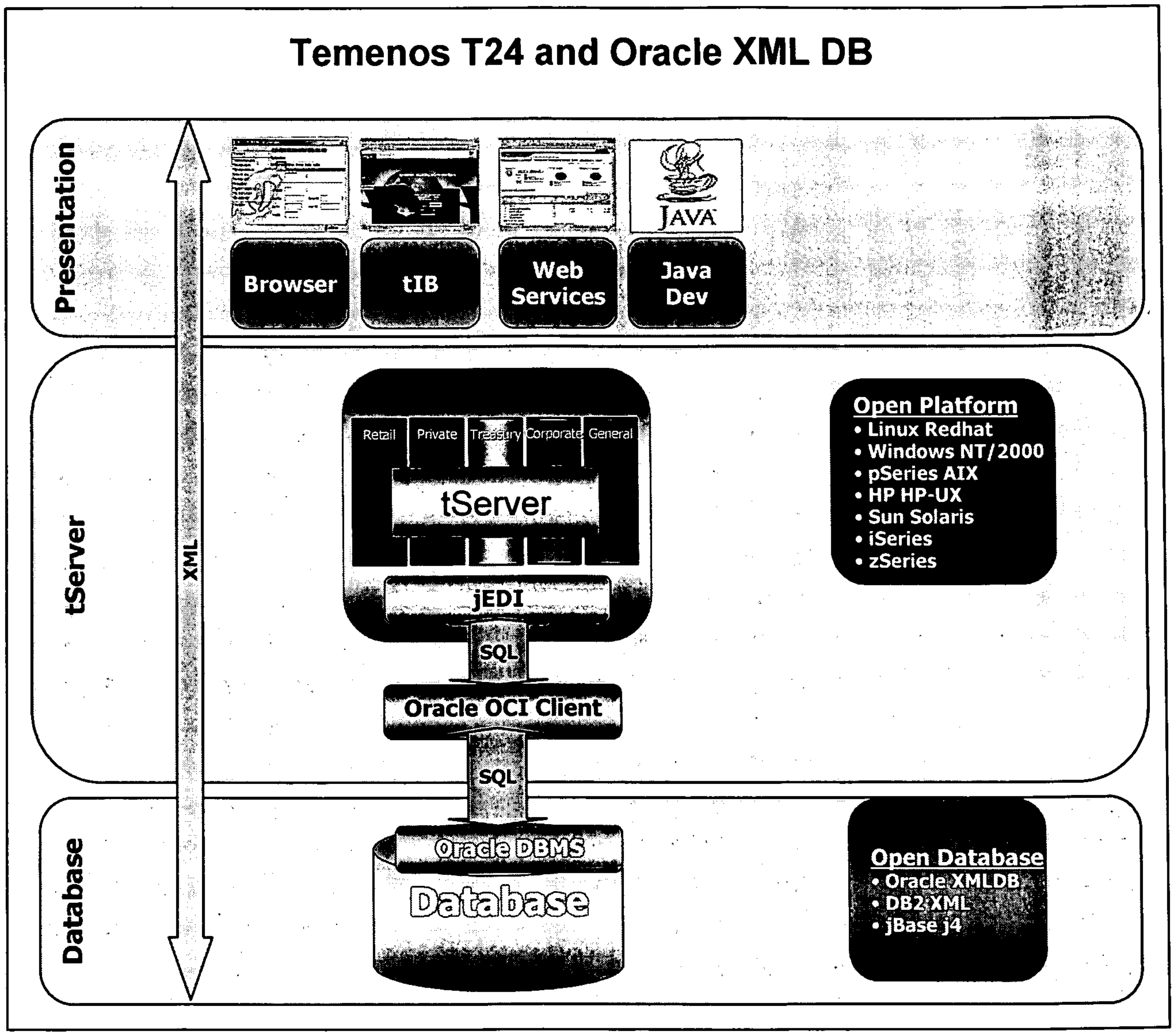

Illustration 2. Temenos T24 relies on the XML/SQL duality of Oracle XML DB. T24 writes XML elements directly into the underlying database. T24 banking applications, however, use SQL to query the database and produce reports.

Temenos uses the most recent version of Oracle Database to manage enterprise data within a single database. T24 can support its multi-value data model as nested sets of XML elements and, at the same time, use SQL for advanced query and reporting capabilities. Relying on the XML/SQL duality, T24 can optimize information management and delivery for enterprise scale operations.

Adapting to Changing Business Needs

Benefits of an Extensible Environment

By using XML for its underlying data storage, T24 provides a platform that can rapidly adapt to changing business requires. Oracle XML DB provides the high-performance, industrial-strength database infrastructure for a competitive banking application.

Let's look at one example that is of particular concern to financial institutions around the world that expect to do business in the United States.

The Problem: New Compliance Requirements

Consider the impact of the USA Patriot Act on a bank's operations and operational application environment. As a result of new banking regulations mandated by this law, a bank's compliance officer requires an immediate change to its banking applications for its retail, private banking, and corporate banking operations. In the future, the bank is to record all ordering parties for all international payments above a predetermined amount. The current system does not record ordering parties on a payment or have any capabilities to adjust to a triggering threshold amount.

This requested change requires modifications to the operational banking application at:

- The front end and integration layer to capture the data
- The application layer to process and validate the data
- The database layer to hold the data

In addition, the bank must produce reports on all international payments and their ordering customers in a format specified by banking regulators.

The Solution: A Platform for Rapid Compliance

With T24 and Oracle XML DB, an application developer only needs to add a new set of repeating group data, defined as XML elements, to the payment transaction to record the ordering party. This change is immediately and automatically propagated to the XML Schema document within Oracle XML DB so that the database can store the new data. This change is also immediately available on the T24 input screens for manual capture, and to the XML Schemas that define the Web Services layer, enabling electronic capture of the required data.

The SQL/XML duality available in Oracle XML DB enables a report of all ordering parties to be made available to the regulator using standard SQL-based reporting tools. In a traditional relational database model (without XML support), producing a new table—as well as capturing and storing the required data—would be required, entailing much more work including, probably, programming efforts.

Consequently, an application developer can use T24 to enhance (and test) a bank's operational system in a matter of minutes rather than days or weeks. A bank can be assured that not only will its enterprise banking application deliver the functionality required for doing business in a competitive, global marketplace, it will also be able to easily adapt to changing regulatory requirements.

Oracle XML DB in Operation: Benchmarking Temenos T24

Workload Scripts that Focus on Performance

The key to successful deployment is not only application functionality, but also system performance.

Oracle and Temenos have worked together in a software testing laboratory to develop, implement, and measure results of a simulated high-end retail banking environment. They developed and tested a retail banking benchmark, based on a representative banking application system, constituting:

- 800,000 customers
- Over 2 million accounts
- Stored in an 18 gigabyte database.

The benchmark script represented a typical banking workload, performing such functions as updating accounts, clearing financial transactions, requesting letters of credit, querying account balances, making deposits, requesting loans, and so forth.

T24 and Oracle XML DB were installed on an HP 9000 Superdome system with 64 processors, running at 875 MHz and encompassing 256 gigabytes RAM. Both the application and the database ran within its own 32 processor, 128 gigabyte partition, and shared access to an XM1024 disk array.

Results

The testers scaled up to 900 simultaneous users within the test environment and measured the system performance results. Each user performed a series of scripted transactions, determined by the benchmark script. The results of this simulated testing environment are shown in Illustration 3.

The test environment was able to support an average of 755.98 application transactions per second, and peaked out at 796.67 application transactions per second. (There are three database transactions for every application transaction.) Processor usage rose to 51 percent for the application server partition and 37 percent for the database server partition. Additional stress testing, reducing the number of processors and the size RAM memory size by half, only slightly degraded the overall performance results.

The simulated benchmark thus demonstrates that T24 running with Oracle XML DB provides a scalable, high-performance enterprise application environment.

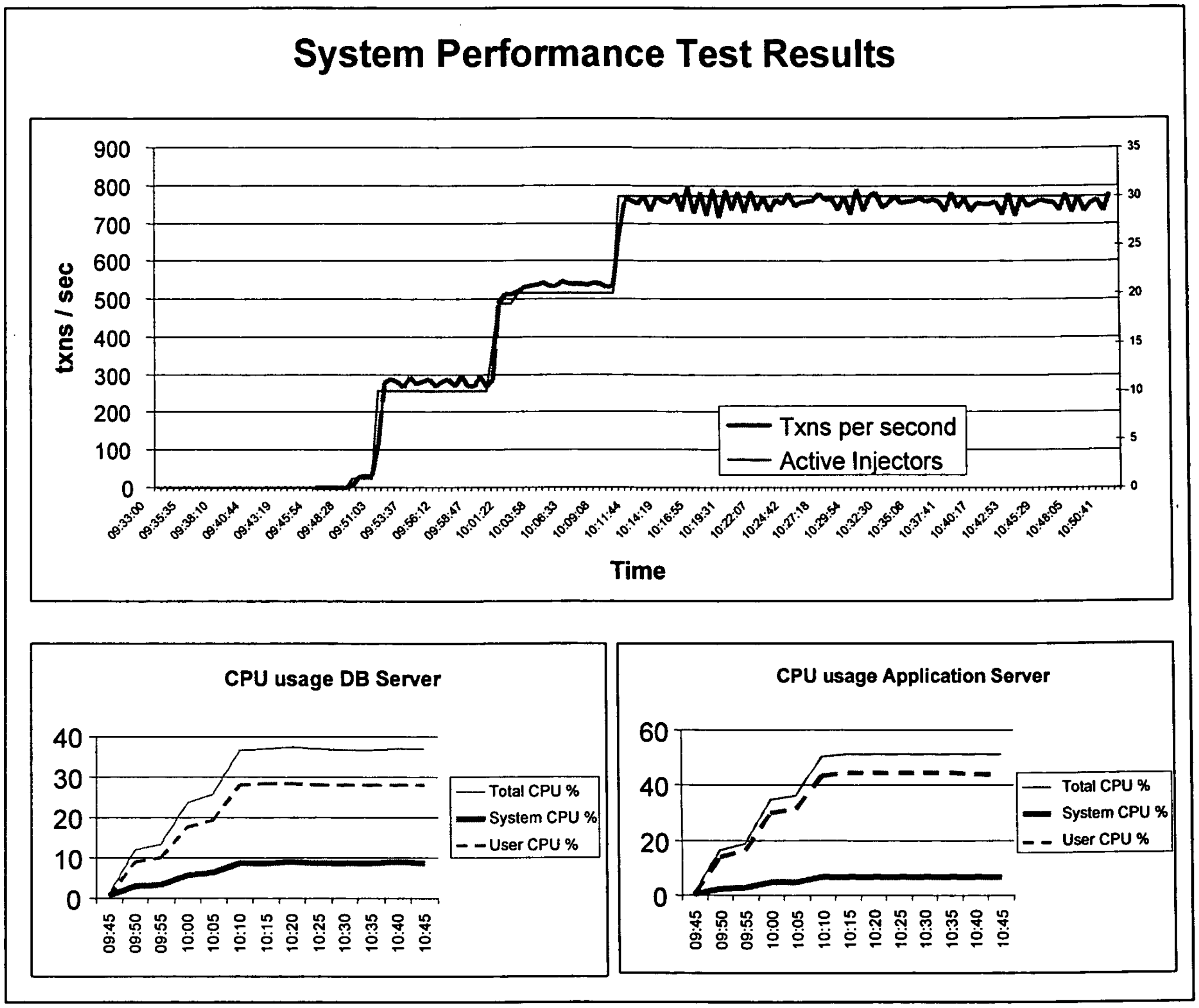

Illustration 3. Temenos T24, running Oracle XML DB that is installed on an HP 9000 system with 64 processors, produces consistent results. The benchmark workload script supports an average of 755.98 application transactions per second. CPU usage reaches a plateau of 40 to 50 percent, depending upon the system performance metric.

The Value Proposition Driving Oracle XML DB

Mainstreaming XML at the Information Layer

In sum, Oracle XML DB provides considerable flexibility for enterprise application design and development without degraded system performance. It demonstrates the enterprise-class scalability of the XML/SQL duality, as well as the commercial viability of an XML database. It mainstreams XML as a native data type within the information layer of an application.

Temenos is now able to deliver a scalable and robust enterprise application that uses the native XML features within the Oracle 9*i* Release 2 and Oracle 10*g* Database. T24 is a highly-competitive banking application based on a multi-value data model which provides unparalleled flexibility for rapid application development. Application developers can build and deploy enterprise applications that reduce the bank's risk of doing business while also increasing the breadth of services the bank can readily provide to its customers.

The Future of XML-Based Enterprise Applications

Temenos, we believe, is only the first of a growing number of ISVs and application development organizations that will reap the business benefits of Oracle XML DB for its enterprise application infrastructure. Through its industry-leading capability to store XML as a native datatype, Oracle is embedding XML tags and XML semantics directly into definitions of individual data elements within a database.

Application developers can now use XML to build and deploy high-performance enterprise applications. They can rapidly adapt these applications to changing business requirements, building on the inherent extensibility of XML. They can manage their data in a systematic manner using XML Schema, and query the underlying repository using XPath and XQuery. Moreover, with the built-in XML/SQL duality within Oracle XML DB, application developers can continue to use SQL for the structured query, access, and data retrieval capabilities for which it was designed.

Oracle XML DB, in short, creates a win/win situation for application developers. They can now build and deploy enterprise applications that use both XML and SQL and can optimize for both the semantic richness of XML and the structured query capabilities of SQL. The result, we believe, is a new generation of enterprise applications, one that encapsulates rich semantic information into strategic, line-of-business operational environments.